



(12)发明专利

(10)授权公告号 CN 104484192 B

(45)授权公告日 2017.11.14

(21)申请号 201510006578.2

(56)对比文件

(22)申请日 2015.01.07

CN 1841328 A, 2006.10.04,

(65)同一申请的已公布的文献号

CN 101937343 A, 2011.01.05,

申请公布号 CN 104484192 A

CN 101256492 A, 2008.09.03,

(43)申请公布日 2015.04.01

CN 101208660 A, 2008.06.25,

(73)专利权人 南威软件股份有限公司

US 7496905 B2, 2009.02.24,

地址 362000 福建省泉州市丰泽区丰海路
南威大厦2号楼16-22层

US 6412105 B1, 2002.06.25,

审查员 王婧阳

(72)发明人 侯济恭

(74)专利代理机构 泉州市文华专利代理有限公司 35205

代理人 陈雪莹

(51)Int.Cl.

G06F 9/44(2006.01)

权利要求书2页 说明书6页 附图3页

G06F 9/45(2006.01)

(54)发明名称

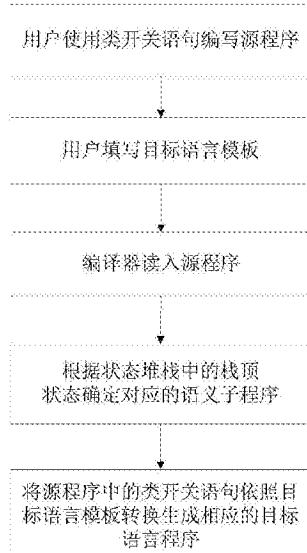
一种复杂多分支结构代码自动生成的方法

(57)摘要

本发明提供一种复杂多分支结构代码自动生成的方法，包括：用户使用类开关语句编写源程序；用户填写目标语言模板；编译器通过总控程序读入所述源程序，并根据状态堆栈中的栈顶状态确定对应的语义子程序，从而将源程序中的类开关语句依照目标语言模板转换生成相应的目标语言程序。本发明的优点在于，用户只需按语法要求书写类开关语句和目标语言模板，编译器即可生成复杂条件下的多分支目标语言程序，且用户修改目标语言模板中的内容，就能改变生成的目标语言程序。本发明解决了多分支程序难于设计、执行效率低、可读性差等问题，改善了代码的可维护性和鲁棒性，实现了复杂条件下多分支结构代码的自动生成。

B

CN 104484192 B



1. 一种复杂多分支结构代码自动生成的方法,其特征在于:所述方法包括如下步骤:

步骤11、用户使用类开关语句编写源程序,所述类开关语句的结构包括:类开关语句头部、类开关语句分支部和类开关语句尾部,其中,类开关语句分支部包括分支条件表达式、分支处理语句及分支结束语句,类开关语句尾部包括缺省处理语句和循环结束语句;

步骤12、用户填写目标语言模板,所述用户在目标语言模板中填写的内容由语言说明、永真循环说明、条件说明、跳出循环说明和终止循环说明构成;

步骤13、编译器通过总控程序读入所述源程序,并根据状态堆栈中的栈顶状态确定对应的语义子程序,从而将源程序中的类开关语句依照所述目标语言模板转换生成相应的目标语言程序;所述编译器包括目标语言模板、总控程序、状态堆栈和语义子程序集,其中,所述语义子程序集由多个用于转换类开关语句的语义子程序组成;

所述的栈顶状态有以下六个状态: S_0 、 S_1 、 S_{11} 、 S_{12} 、 S_{error} 和 S_{end} ,相应的,语义子程序集也包括六个语义子程序为 P_0 、 P_1 、 P_{11} 、 P_{12} 、 P_{error} 和 P_{end} ;对各栈顶状态定义如下:

S_0 :为初始状态,当读入的是非类开关语句,则原样输出并保持原状态 S_0 ;当读入的是类开关语句头部“switchH”,则转入 S_1 状态;

S_1 :用于类开关语句头部处理:当读入的是类开关语句分支部“caseH”,转入 S_{11} 状态;当读入的是类开关语句尾部“defaultH”,则转入 S_{12} 状态;

S_{11} :用于类开关语句分支部处理;当读入的是分支结束语句“breakH”时,表明该分支结束,状态变为 S_1 ;否则,输出源程序语句;

S_{12} :用于类开关语句尾部处理,当读入的是循环结束语句“breakH”时,表明循环结束,状态变为 S_0 ;

S_{error} :用于错误处理;

S_{end} :为源程序处理完毕状态,用于判断源程序是否终止;

若此时的栈顶状态为 S_0 ,则确定相应的语义子程序为 P_0 ,执行以下步骤:

- (1) 判断源程序语句是否是类开关语句头部switchH:若是,则跳转(2),否则跳转(4);
- (2) 根据目标语言模板,生成永真循环“while(1){};
- (3) S_0 退出状态堆栈,状态 S_1 进栈,跳转(5);
- (4) 输出源程序语句;
- (5) 返回主程序;

若此时的栈顶状态为 S_1 ,则确定相应的语义子程序为 P_1 ,执行以下步骤:

- (1) 判断源程序语句是否是类开关语句分支部caseH:若是,则跳转(2),否则跳转(4);
- (2) 根据目标语言模板,生成条件语句“if(条件式){};
- (3) S_1 退出状态堆栈,状态 S_{11} 进栈,跳转(7);
- (4) 判断源程序语句是否是类开关语句尾部defaultH:若是,则跳转(5),否则跳转(6);
- (5) S_1 退出状态堆栈,状态 S_{12} 进栈,跳转(7);
- (6) 输出源程序语句;
- (7) 返回主程序;

若此时的栈顶状态为 S_{11} ,则确定相应的语义子程序为 P_{11} ,执行以下步骤:

- (1) 判断源程序语句是否是分支结束语句breakH:若是,则跳转(2),否则跳转(4);
- (2) 根据目标语言模板,生成跳出循环语句“break;{}”;

(3) S₁₁退出状态堆栈,状态S₁进栈,跳转(5);

(4) 输出源程序语句;

(5) 返回主程序;

若此时的栈顶状态为S₁₂,则确定相应的语义子程序为P₁₂,执行以下步骤:

(1) 判断源程序语句是否是循环结束语句breakH:若是,则跳转(2),否则跳转(4);

(2) 根据目标语言模板,生成终止循环语句“break;”;

(3) S₁₂退出状态堆栈,状态S₀进栈,跳转(5);

(4) 输出源程序语句;

(5) 返回主程序;

若此时的栈顶状态为S_{error},则确定相应的语义子程序为P_{error}:标注源程序中的错误语句;

若此时的栈顶状态为S_{end},则确定相应的语义子程序为P_{end}:源程序终止,退出编译。

2. 根据权利要求1所述的一种复杂多分支结构代码自动生成的方法,其特征在于:所述编译器执行的编译过程具体为:

步骤21、读入目标语言模板;

步骤22、初始状态进栈;

步骤23、读入源程序,开始编译;

步骤24、判断源程序是否终止:若是,则退出编译;若否,则跳转步骤25;

步骤25、读状态堆栈当前的栈顶状态;

步骤26、根据当前的栈顶状态确定对应的语义子程序,进行类开关语句的转换,生成相应的目标语言程序并修改栈顶状态;

步骤27、返回步骤23。

3. 根据权利要求1或2所述的一种复杂多分支结构代码自动生成的方法,其特征在于:所述用户修改目标语言模板中的内容,所述编译器将对应更换生成的目标语言程序。

一种复杂多分支结构代码自动生成的方法

技术领域

[0001] 本发明涉及计算机软件编译系统领域,更具体地说,涉及一种复杂多分支结构代码自动生成的方法。

背景技术

[0002] 目前,在C++、C#或Java编程中,对于多分支的业务需求,只能用开关语句(switch-case)或条件判断语句(if-then-else)来解决。开关语句的选择因子只能是常量,因此对复杂的条件判断无能为力。用条件判断语句能解决复杂条件的多分支结构,但是,只能运用多重嵌套形式完成。

[0003] 以C++为例,复杂条件的多分支程序设计:判断读入的字符类型,其典型范例是:

```
if ( inputCharacter <SPACE ) {  
    CharacterType = CharacterType_ControlCharacter;  
}  
  
else if ( inputCharacter == ' ' || inputCharacter == ',', ',' ||  
         inputCharacter == '.', '.' || inputCharacter == '!','!' ||  
         inputCharacter == '(' || inputCharacter == ')',')' ||  
         inputCharacter == ';' ';' || inputCharacter == ';',';' ||  
         inputCharacter == '?' '?' || inputCharacter == '_','_')  
{  
    CharacterType = CharacterType_punctuation;  
}  
  
else if ( '0' <= inputCharacter && inputCharacter <= '9' ) {  
    CharacterType = CharacterType_Digit;
```

```
    }  
    else if ( ('a' <= inputCharacter && inputCharacter <= 'z') ||  
[0005]    ('A' <= inputCharacter && inputCharacter <= 'Z')  
    ) {  
    CharacterType = CharacterType_Letter  
}
```

[0006] 这种结构,随着条件判断分支的增加,嵌套也随之增加,从而导致句式冗长,可读性很差且调试困难,程序的鲁棒性难以得到保证。

[0007] 所以,本发明人对复杂条件的多分支结构程序设计进行了深入研究,由此产生本案。

发明内容

[0008] 本发明要解决的技术问题,在于提供一种复杂多分支结构代码自动生成的方法,通过提供一种类开关语句的结构和编译器,用户只需按语法要求书写类开关语句和目标语言模板,编译器即可生成复杂条件下的多分支目标语言程序,且用户修改目标语言模板中的内容,就能改变生成的目标语言程序。解决了多分支程序难于设计、执行效率低、可读性差等问题,改善了代码的可维护性和鲁棒性,实现了复杂条件下多分支结构代码的自动生成。

[0009] 本发明是这样实现的:一种复杂多分支结构代码自动生成的方法,包括如下步骤:

[0010] 步骤11、用户使用类开关语句编写源程序,所述类开关语句的结构包括:类开关语句头部、类开关语句分支部和类开关语句尾部,其中,类开关语句分支部包括分支条件表达式、分支处理语句及分支结束语句,类开关语句尾部包括缺省处理语句和循环结束语句;

[0011] 步骤12、用户填写目标语言模板,所述目标语言模板由语言说明、永真循环说明、条件说明、跳出循环说明和终止循环说明构成;

[0012] 步骤13、编译器通过总控程序读入所述源程序,并根据状态堆栈中的栈顶状态确定对应的语义子程序,从而将源程序中的类开关语句依照所述目标语言模板转换生成相应的目标语言程序;所述编译器包括目标语言模板、总控程序、状态堆栈和语义子程序集,其中,所述语义子程序集由复数个用于转换类开关语句的语义子程序组成。

[0013] 进一步的,所述编译器执行的编译过程具体为:

[0014] 步骤21、读入目标语言模板;

[0015] 步骤22、初始状态进栈;

[0016] 步骤23、读入源程序,开始编译;

[0017] 步骤24、判断源程序是否终止:若是,则退出编译;若否,则跳转步骤25;

[0018] 步骤25、读状态堆栈当前的栈顶状态;

[0019] 步骤26、根据当前的栈顶状态确定对应的语义子程序,进行类开关语句的转换,生

成相应的目标语言程序并修改栈顶状态；

[0020] 步骤27、返回步骤23。

[0021] 进一步的，所述用户修改目标语言模板中的内容，所述编译器将对应更换生成的目标语言程序。

[0022] 采用上述方案后，本发明具有如下优点：

[0023] 1、通过提供一种类开关语句的结构，利用开关语句头部实现循环结构、开关语句分支部实现各分支条件和跳出循环的判断，开关语句尾部实现循环的终止，从而完成复杂多分支语句的程序设计；

[0024] 2、依照目标语言模板生成的目标语言程序，每一个条件分支处理完毕，即刻跳出循环，同时，在整个循环的终点，执行缺省处理后也跳出循环，终止语句，既保证了程序不会陷入死循环之中，又提高了程序的执行效率；

[0025] 3、编译器通过执行总控程序的算法，可快速生成目标语言程序，提高复杂多分支程序设计的效率，并极大改善代码的可维护性和鲁棒性；

[0026] 4、用户只需改变目标语言模板的内容，编译器即可生成相应的目标语言程序，适用性高，可以广泛应用于C++，Java，C#等程序设计语言。

附图说明

[0027] 下面参照附图结合实施例对本发明作进一步的说明。

[0028] 图1为本发明方法执行流程图。

[0029] 图2为本发明方法一实施例的目标语言程序结构示意图。

[0030] 图3为本发明方法一实施例的编译过程流程图。

具体实施方式

[0031] 请参阅图1，本发明，一种复杂多分支结构代码自动生成的方法，包括如下步骤：

[0032] 步骤11、用户使用类开关语句编写源程序，所述类开关语句的结构包括：类开关语句头部、类开关语句分支部和类开关语句尾部，其中，类开关语句分支部包括分支条件表达式、分支处理语句及分支结束语句，类开关语句尾部包括缺省处理语句和循环结束语句；

[0033] 以一类开关语句为例，其结构如下所示：

switchH //类开关语句头部

 caseH 条件式 1 //分支条件表达式

 语句串 1; //分支处理语句

[0034] breakH; //分支结束语句

.....

 defaultH: 语句串 n //缺省处理语句

 breakH; //循环结束语句

[0035] 步骤12、用户填写目标语言模板，所述目标语言模板由语言说明、永真循环说明、

条件说明、跳出循环说明和终止循环说明构成；

[0036] 例如，以C++为目标语言的的目标语言模板说明如下：

```
C++          //语言说明
while(1){}    //循环说明
[0037] if(){}      //条件语句说明
break        // 跳出循环说明
Default: break //终止循环说明
```

[0038] 步骤13、编译器通过总控程序读入所述源程序，并根据状态堆栈中的栈顶状态确定对应的语义子程序，从而将源程序中的类开关语句依照所述目标语言模板转换生成相应的目标语言程序；所述编译器包括目标语言模板、总控程序、状态堆栈和语义子程序集，其中，所述语义子程序集由复数个用于转换类开关语句的语义子程序组成。例如，有以下六个状态： $S_0, S_1, S_{11}, S_{12}, S_{\text{error}}$ 和 S_{end} ，相应的，语义子程序集也包括六个语义子程序为 $P_0, P_1, P_{11}, P_{12}, P_{\text{error}}$ 和 P_{end} ；可对各状态定义如下：

[0039] S_0 ：为初始状态，当读入的是非类开关语句，则原样输出并保持原状态 S_0 ；当读入的是类开关语句头部“switchH”，则转入 S_1 状态；

[0040] S_1 ：用于类开关语句头部处理：当读入的是类开关语句分支部“caseH”，转入 S_{11} 状态；当读入的是类开关语句尾部“defaultH”，则转入 S_{12} 状态；

[0041] S_{11} ：用于类开关语句分支部处理；当读入的是分支结束语句“breakH”时，表明该分支结束，状态变为 S_1 ；否则，输出源程序语句；

[0042] S_{12} ：用于类关开语句尾部处理，当读入的是循环结束语句“breakH”时，表明循环结束，状态变为 S_0 ；

[0043] S_{error} ：用于错误处理；

[0044] S_{end} ：为源程序处理完毕状态，用于判断源程序是否终止；

[0045] 以上述C++目标语言模板为例，则编译器生成的目标语言程序的结构如图2所示：

```
while(1)
{
    if(条件1) {语句串1; break;}
    if(条件2) {语句串2; break;}
    .....
    if(条件n) {语句串n; break;}
    default: 语句串n+1;
    break;
}
```

[0047] 在循环体中，若条件1满足，则执行语句串1，并跳出循环；若条件2满足，则执行语

句串2，并跳出循环，如此顺序执行，直至各个分支完成；终止循环处理即缺省处理：执行语句串n+1，并跳出循环；依照目标语言模板生成的目标语言程序，每一个条件分支处理完毕，即刻跳出循环，同时，在整个循环的终点，执行缺省处理后也跳出循环，终止语句，既保证了程序不会陷入死循环之中，又提高了程序的执行效率。

[0048] 通过步骤11到步骤13，提供一种类开关语句的结构，实现复杂多分支语句的程序设计，同时由编译器执行总控程序的算法，依据目标语言模板快速生成目标语言程序，提高了复杂多分支程序设计的效率，极大改善代码的可维护性和鲁棒性。

[0049] 如图3所示，所述编译器执行的编译过程具体为：

[0050] 步骤21、读入目标语言模板；

[0051] 步骤22、初始状态进栈；

[0052] 步骤23、读入源程序，开始编译；

[0053] 步骤24、判断源程序是否终止：若是，则退出编译；若否，则跳转步骤25；

[0054] 步骤25、读状态堆栈当前的栈顶状态；

[0055] 步骤26、根据当前的栈顶状态确定对应的语义子程序，进行类开关语句的转换，生成相应的目标语言程序并修改栈顶状态；

[0056] 步骤27、返回步骤23；

[0057] 例如，若此时的栈顶状态为S₀，则确定相应的语义子程序为P₀，执行以下步骤：

[0058] (1)：判断源程序语句是否是类开关语句头部switchH：若是，则跳转(2)，否则跳转(4)；

[0059] (2)：根据目标语言模板，生成永真循环“while (1) {”；

[0060] (3)：S₀退出状态堆栈，状态S₁进栈，跳转(5)；

[0061] (4)：输出源程序语句；

[0062] (5)：返回主程序；

[0063] 若此时的栈顶状态为S₁，则确定相应的语义子程序为P₁，执行以下步骤：

[0064] (1)：判断源程序语句是否是类开关语句分支部caseH：若是，则跳转(2)，否则跳转(4)；

[0065] (2)：根据目标语言模板，生成条件语句“if (条件式) {”；

[0066] (3)：S₁退出状态堆栈，状态S₁₁进栈，跳转(7)；

[0067] (4)：判断源程序语句是否是类开关语句尾部defaultH：若是，则跳转(5)，否则跳转(6)；

[0068] (5)：S₁退出状态堆栈，状态S₁₂进栈，跳转(7)；

[0069] (6)：输出源程序语句；

[0070] (7)：返回主程序；

[0071] 若此时的栈顶状态为S₁₁，则确定相应的语义子程序为P₁₁，执行以下步骤：

[0072] (1)：判断源程序语句是否是分支结束语句breakH：若是，则跳转(2)，否则跳转(4)；

[0073] (2)：根据目标语言模板，生成跳出循环语句“break; }”；

[0074] (3)：S₁₁退出状态堆栈，状态S₁进栈，跳转(5)；

[0075] (4)：输出源程序语句；

- [0076] (5) :返回主程序；
- [0077] 若此时的栈顶状态为 S_{12} ,则确定相应的语义子程序为 P_{12} ,执行以下步骤：
- [0078] (1) :判断源程序语句是否是循环结束语句breakH:若是,则跳转(2),否则跳转(4)；
- [0079] (2) :根据目标语言模板,生成终止循环语句“break;”；
- [0080] (3) : S_{12} 退出状态堆栈,状态 S_0 进栈,跳转(5)；
- [0081] (4) :输出源程序语句；
- [0082] (5) :返回主程序；
- [0083] 若此时的栈顶状态为 S_{error} ,则确定相应的语义子程序为 P_{error} :标注源程序中的错误语句；
- [0084] 若此时的栈顶状态为 S_{end} ,则确定相应的语义子程序为 P_{end} :源程序终止,退出编译。
- [0085] 所述用户修改目标语言模板中的内容,所述编译器将对应更换生成的目标语言程序;这样用户只需改变目标语言模板的内容,编译器即可生成相应的目标语言程序,适用性高,可以广泛应用于C++,Java,C#等程序设计语言。
- [0086] 本发明通过提供一种类开关语句的结构和编译器,用户只需按语法要求书写类开关语句和目标语言模板,编译器即可生成复杂条件下的多分支目标语言程序,且用户修改目标语言模板中的内容,就能改变生成的目标语言程序。本发明解决了多分支程序难于设计、执行效率低、可读性差等问题,改善了代码的可维护性和鲁棒性,实现了复杂条件下多分支结构代码的自动生成。
- [0087] 虽然以上描述了本发明的具体实施方式,但是熟悉本技术领域的技术人员应当理解,我们所描述的具体的实施例只是说明性的,而不是用于对本发明的范围的限定,熟悉本领域的技术人员在依照本发明的精神所作的等效的修饰以及变化,都应当涵盖在本发明的权利要求所保护的范围内。

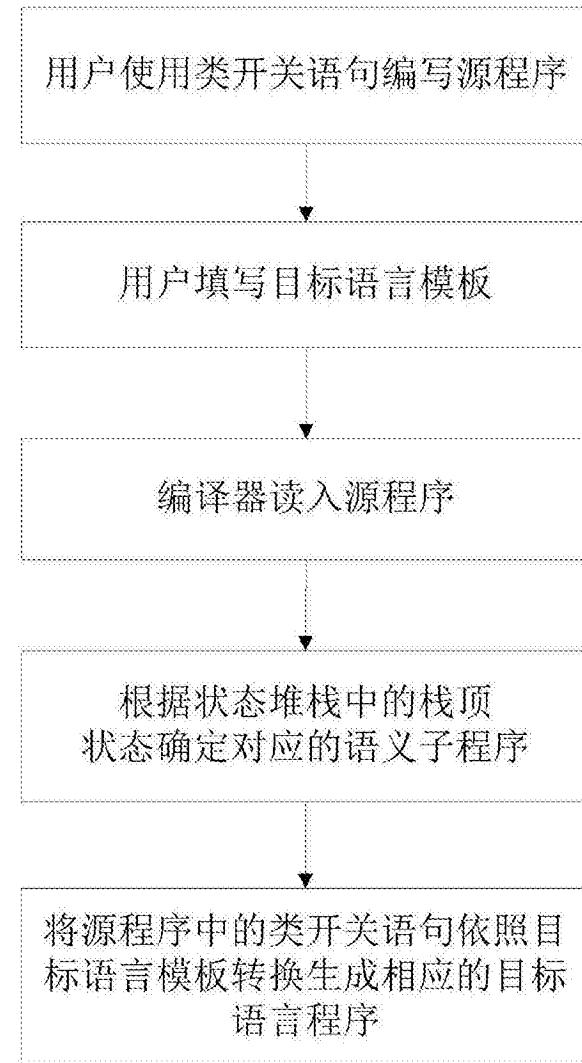


图1

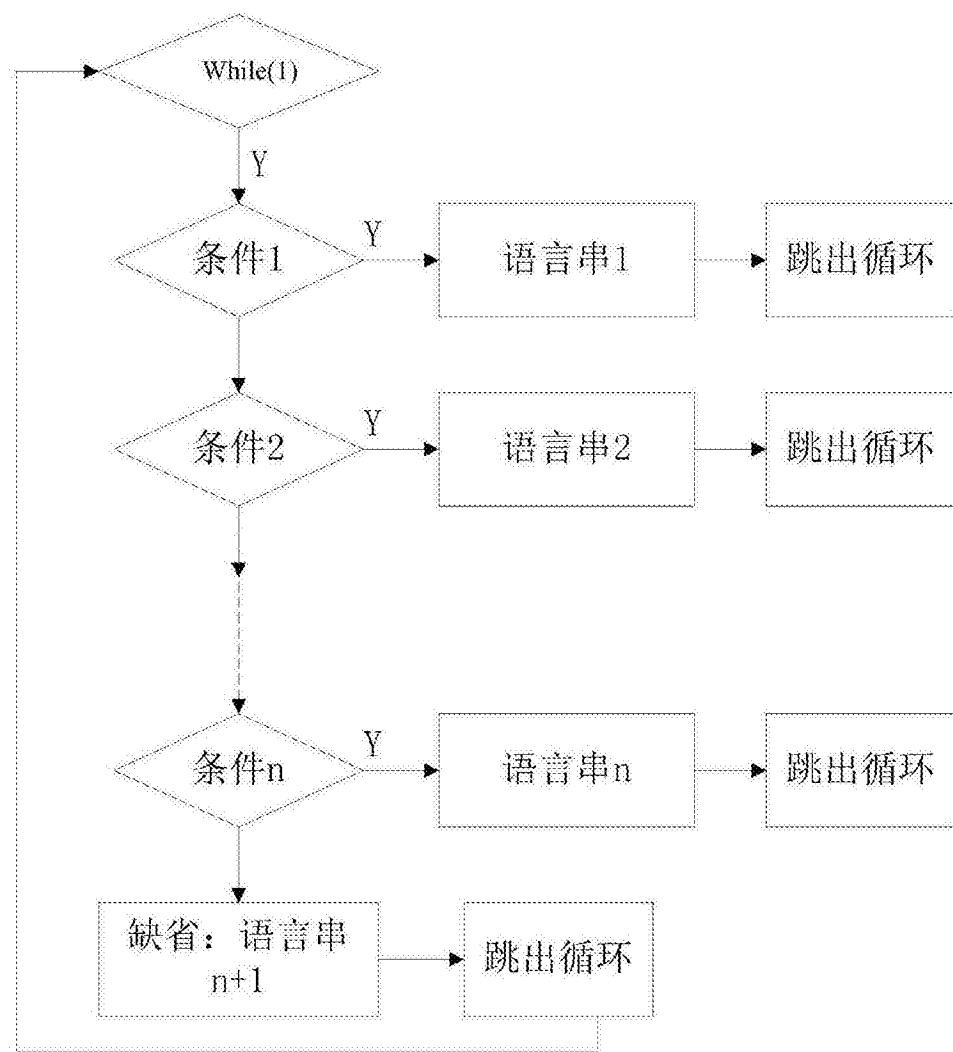


图2

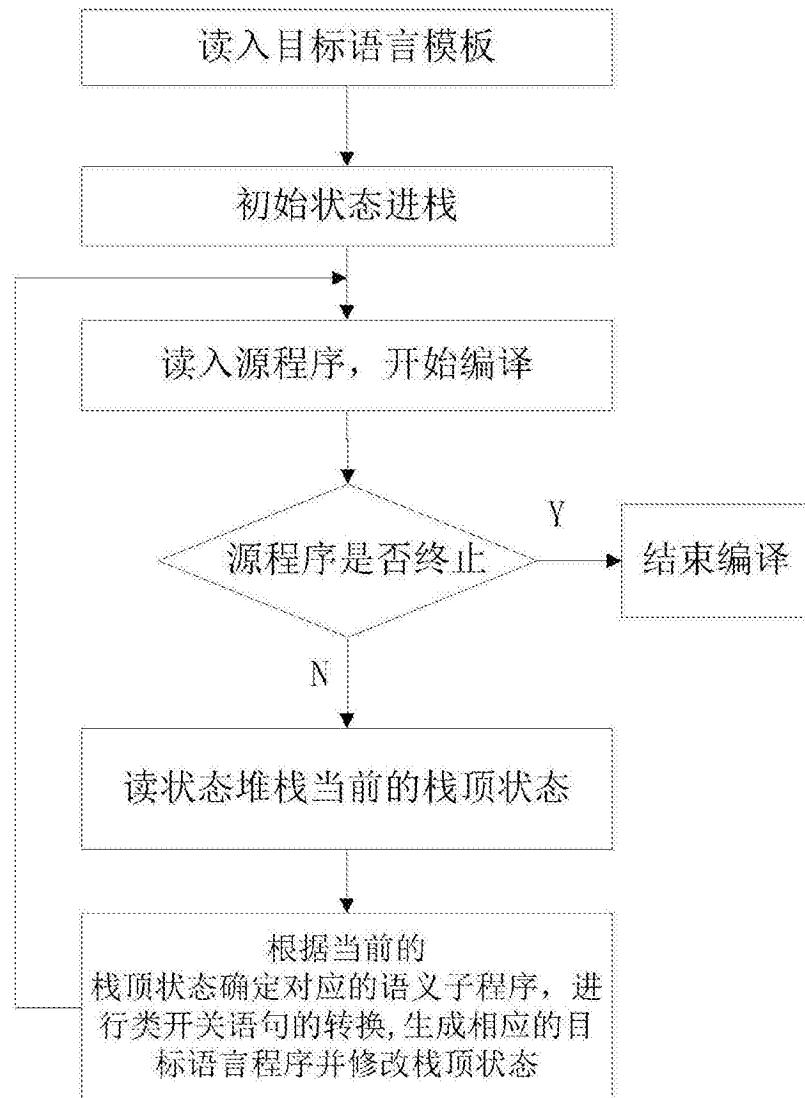


图3